The Hypercube

- A d-dimensional hypercube consists of 2^d nodes
  - Each of the 2^d nodes is labeled with a unique d-bit binary string $a_1 \ldots a_d$ called the ID of the node
  - Each node is directly connected to the d nodes whose IDs differ in exactly one bit position
  - For each pair of adjacent nodes u and v, we will assume that there are two directed edges (u, v) and (v, u)
  - Thus the total number of directed edges is $d2^d$
  - A "dimension-i" edge connects two nodes whose IDs differ in bit position i, $1 \leq i \leq d$

Routing a Single Packet on the Hypercube

- Suppose we have a parallel computer with the topology of a d-dimensional hypercube
  - Each of the 2^d nodes corresponds to a processor with an unbounded local memory
  - Each of the $d2^d$ edges corresponds to a directed communication channel
  - The processors communicate with one another by sending "packets" over the communication channels
  - Each packet holds $O(1)$ machine words of information
  - In each "time step" we allow each directed edge to transmit one packet
- How can we route a packet from source $a_1 \ldots a_d$ to destination $b_1 \ldots b_d$?

The Bit-Fixing Routing Algorithm

- The "bit-fixing" routing algorithm "corrects bits" in increasing order of index to route a packet from source $a_1 \dots a_d$ to destination $b_1 \dots b_d$
  - Let I denote $\{i \in \{1, \dots, d\} \mid a_i \neq b_i\}$
  - Index the elements of I as $i_1 < \dots < i_k$ where $k \leq d$
  - The packet first takes the dimension-i_1 edge out of the source to "correct" bit i_1 and arrive at some node α
  - Then the packet follows the dimension-i_2 edge out of node α to correct bit i_2, et cetera
  - After k "hops", the packet arrives at its destination

Permutation Routing on the Hypercube

- In a permutation routing problem, each node is the source of exactly one packet, and each node is the destination of exactly one packet
- The bit-fixing approach for routing a single packet suggests a natural permutation routing algorithm
  - Each packet moves along the bit-fixing path from its source to its destination
  - Packets waiting to traverse a given outgoing edge are stored in an associated (unbounded) queue
  - A queueing discipline (e.g., FIFO) is used to determine which of the packets in a given edge queue gets to advance in each time step

The Performance of Bit-Fixing Permutation Routing

- Is there a queueing discipline with respect to which the bit-fixing algorithm routes an arbitrary permutation in $O(d)$ time steps?
- Unfortunately, the answer is no
  - Suppose $d = 2k$, and consider the "transpose" permutation that routes the packet at any given source $a_1 \ldots a_{2k}$ to destination $a_{k+1} \ldots a_{2k} a_1 \ldots a_k$
  - Let X denote the set of 2^{k-1} nodes $a_1 \ldots a_{2k}$ where $a_k = 1$ and $a_i = 0$ for $k < i \leq 2k$
  - Each packet with a source in X traverses the dimension-k edge into the all-zeros node
  - Thus, $|X| = 2^{k-1} = \Omega(2^{d/2})$ is a lower bound on the number of time steps required to route the transpose permutation

Valiant's Two-Phase Permutation Routing Algorithm

- In the first phase, each source independently picks a uniformly random node as an intermediate destination for its packet, and the bit-fixing scheme is used to route each packet to its intermediate destination
  - We will focus on proving that the first phase terminates within $O(d)$ steps with high probability
- In the second phase, the bit-fixing scheme is used to route all of the packets from their intermediate destinations to their final destinations
  - Since the second phase corresponds to "running the first phase in reverse", our analysis of the first phase implies the same time bound for the second phase

Some Remarks

- Our analysis holds for any "greedy" queueing discipline: Whenever an edge queue is nonempty, some packet is allowed to advance
- It is easy to enforce a synchronization barrier between the two phases at the cost of $O(d)$ additional time steps, but it is not necessary to do so
  - It is more natural to allow any packet that reaches its intermediate destination to immediately begin moving towards its final destination
  - It is not difficult to extend our analysis to handle this variant

Analysis of the First Phase: Preliminaries

- For any node α, we refer to the packet with source α as packet α, and we refer to the bit-fixing path traversed by packet α in the first phase as path α
- Fix an arbitrary node β
  - We number the time steps from 1, and we let T denote the number of time steps required for packet β to reach its intermediate destination
  - We will prove that $T = O(d)$ with high probability
- Let P denote path β, and let the sequence of directed edges on P be $e_1, \ldots, e_k$
  - If $k = 0$ then $T = 0$
  - In what follows, we assume that $k > 0$

White, Gray, and Black Packets

- At the start of each time step, we assign a color to each packet α as follows
  - If the suffix of path α that remains to be traversed by packet α does not include any edges on P, then packet α is black
  - If packet α is in the queue associated with some edge of P, then packet α is gray
  - Otherwise, packet α is white

Characterizing the Possible Color Transitions

- The following three claims are straightforward to prove
  - Remark: We will not need to use Lemma 1
- Lemma 1: If a packet is white at the start of time step t, then it is white or gray at the start of time step $t + 1$
- Lemma 2: If a packet is gray at the start of time step t, then it is gray or black at the start of time step $t + 1$
- Lemma 3: If a packet is black at the start of time step t, then it is black at the start of time step $t + 1$

The "Lag" of a Gray Packet

- At the start of any time step t, we define the lag of any gray packet α as follows
  - Let j be the unique integer such that packet α is in the queue associated with directed edge e_j
  - Then we define the lag of packet α as $t - j$
- Packet β is gray at the start of each time step t in $\{1, \ldots, T\}$
- The lag of packet β at the start of a time step t in $\{1, \ldots, T\}$ is equal to the number of time steps $t' < t$ such that packet β does not advance at time step t'

Some Useful Events

- Let S denote the set of all nodes α such that $\alpha \neq \beta$ and path α shares at least one directed edge with path P
- For any positive integer i and time step t, let $A_{i,t}$ denote the event that packet β has lag i (resp., $i-1$) at the start of time step t (resp., $t-1$)
- For any nonnegative integer i and time step t, let $B_{i,t}$ denote the event that there exists a node α in S such that packet α has lag i at the start of time step t
- For any nonnegative integer i, let C_i denote the event that there exists a node α in S and a time step t such that packet α has lag i at the start of time step t, and is black at the start of time step $t+1$

If $A_{i+1,t+1}$ Occurs, Then $B_{i,t}$ Occurs

- Lemma 4: For any nonnegative integer i and any time step t, $A_{i+1,t+1}$ implies $B_{i,t}$
  - Assume event $A_{i+1,t+1}$ occurs
  - Thus packet β has lag i at the start of time step t, and packet β has lag $i+1$ at the start of time step $t+1$
  - During time step t, some packet $\alpha \neq \beta$ advances from the edge queue containing packet β
  - Thus node α belongs to S and packet α has lag i at the start of time step t
  - Thus event $B_{i,t}$ occurs

If $A_{i+1,t+1}$ Occurs, Then C_i Occurs

- Lemma 5: For any nonnegative integer i, $A_{i+1,t+1}$ implies C_i
  - Assume event $A_{i+1,t+1}$ occurs
  - Thus Lemma 4 implies that event $B_{i,t}$ occurs
  - Let $t' \geq t$ denote the maximum time step such that event $B_{i,t'}$ occurs
  - There is a node α in S such that at the start of time step t', packet α has lag i and is in the edge queue of some directed edge e_j of P
  - Some packet α' (which could be α) advances from this queue during time step t'
  - The lags of packets α and α' are each equal to i at the start of time step t'

If $A_{i+1,t+1}$ Occurs, Then C_i Occurs (cont'd)

- We claim that α' is not equal to β
  - Assume for the sake of contradiction that $\alpha' = \beta$
  - Since event $A_{i+1,t+1}$ occurs, $t' \geq t$, the lag of packet α' is i at the start of time step t, and the lag of packet β cannot decrease, we deduce that $t' = t$
  - Thus packet α' advances during time step t
  - Since event $A_{i+1,t+1}$ occurs, packet β does not advance during time step t
  - Thus $\alpha' \neq \beta$, a contradiction

If $A_{i+1,t+1}$ Occurs, Then C_i Occurs (cont'd)

- Since $\alpha' \neq \beta$, we deduce that α' belongs to S
- If packet α' is black at the start of time step $t' + 1$, then event C_i occurs, as required
- Otherwise, Lemma 2 implies that packet α' is gray at the start of time step $t' + 1$
  - We deduce that packet α' belongs to the queue of edge e_{j+1} and has lag i at the start of time step $t' + 1$
  - Hence event $B_{i,t'+1}$ occurs, contradicting the definition of t'

An Upper Bound on T

- Lemma 6: $T \leq k + |S|$
  - Since packet β is gray and has lag $T - k$ at the start of time step T, there exist time steps $t_1 < \cdots < t_{T-k}$ such that event $\cap_{1 \leq i \leq T-k} A_{i,t_i}$ occurs
  - Lemma 5 implies that event $\cap_{0 \leq i < T-k} C_i$ occurs
  - Lemma 3 implies that $|S| \geq T - k$
- Since $k \leq d$, it remains to prove that $|S|$ is $O(d)$ with high probability

A Plan for Upper Bounding $|S|$

- Each path α depends on the uniformly random choice of the intermediate destination
- Imagine that we reveal path β (i.e., P) first
- For each node α not equal to β, let the indicator random variable X_α be equal to 1 if path α belongs to S, and 0 otherwise
  - Note that these random variables are independent
  - Let the random variable X denote their sum, which is $|S|$
- Plan: Derive an upper bound on $E(X)$, and then apply a Chernoff-type inequality to obtain a high probability upper bound on X

Upper Bounding $E(X)$

- Let us partition the $2^d - 1$ nodes α that are not equal to β into d groups as follows
    - Let α be a node that is not equal to β
    - Let ℓ be the length of the longest common suffix of α and β
    - Then α belongs to group ℓ, $0 \leq \ell < d$
    - Group ℓ contains $2^{d-\ell-1}$ nodes
- For each of the 2^{d-1} nodes α in group 0, we have $\mathrm{E}(X_\alpha) = 0$

Upper Bounding $E(X)$ (cont'd)

- For each of the $2^{d-\ell-1}$ nodes α in group ℓ, $1 \le \ell < d$, we have $\mathrm{E}(X_\alpha) \le 2^{\ell-d-1}$
  - Let α be a node in group ℓ, $1 \le \ell < d$
  - If every edge of P has dimension at most $d-\ell$, we are assured that $X_\alpha = 0$
  - Otherwise, let i denote the minimum dimension of any edge of P that exceeds $d-\ell$
  - In order for the event $X_\alpha = 1$ to occur, the intermediate destinations of α and β need to match in the first $i \ge d-\ell+1$ bit positions
- Thus

$$\mathrm{E}(X) \le \sum_{1 \le \ell < d} 2^{d-\ell-1} 2^{\ell-d-1} = \frac{d-1}{4} \le \frac{d}{4}$$

A Tail Bound for $X = |S|$

- We can view X as the sum of $n = 2^d$ independent trials with average success probability $p \leq \frac{d}{4n}$
- Using the large deviation Chernoff bound from the previous lecture, we find that

$$\Pr(X \geq (1+\delta)d/4) \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{d/4}$$

  for any $\delta > 0$
- For any positive constant c, we can drive the RHS below n^{-c} by choosing δ to be a sufficiently large positive constant

An Upper Bound for Phase One

- Let c be an arbitrary positive constant
- Combining the previous tail bound with $X = |S|$ and Lemma 6, we find that there is a positive constant c' such that packet β reaches its intermediate destination within $c'd$ steps with probability at least $1 - n^{-c}$
- For any node α, let E_α denote the event that packet α takes more than $c'd$ steps to reach its intermediate destination
- Since β was chosen arbitrarily, we have $\Pr(E_\alpha) \le n^{-c}$ for all nodes α
- By a union bound, we conclude that phase one terminates within $c'd$ steps with probability at least $1 - n^{1-c}$